

Advanced Graphics Package Supplement for HP-UX Systems



HEWLETT-PACKARD COMPANY
Engineering Productivity Division
19420 Homestead Road
Cupertino, California 95014

Update 2 June 1984
MANUAL PART NO. 97085-90001
Printed in U.S.A. June 1983
U0684

Printing History

The Printing History below identifies the Edition of this Manual and any Updates that are included. Periodically, Update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this Printing History page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past Updates. However, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all Updates.

First Edition June 1983
Update 1 November 1983
Update 2 June 1984 (includes Update 1)

"Copyright THE REGENTS OF THE UNIVERSITY OF COLORADO, a body corporate, 1979. ALL RIGHTS RESERVED. This document has been reprinted with the permission of the Regents of the University of Colorado, a body corporate."

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

AGP SUBROUTINES AND PARAMETERS FOR HP-UX SYSTEMS

J2DRW (X,Y)	JIWS (ID,OPCODE,SSIZE,ISIZE,RSIZE, <u>SLIST</u> , <u>ILIST</u> , <u>RLIST</u>)
J2MOV (X,Y)	JJUST (BASE,UP)
J2MRK (X,Y,MARKNO)	JKYBD (ID,ECHO,MAX, <u>ACTUAL</u> , <u>STRING</u>)
J2PGN (NPOLY,NVEC,XVEC,YVEC)	JLLIM (ID,XMIN,XMAX,YMIN,YMAX)
J2PLY (NPNTS,XVEC,YVEC)	JLOCP (ID,VX,VY)
J3DRW (X,Y,Z)	JLPMM (ID,VX,VY, <u>XMM</u> , <u>YMM</u>)
J3MOV (X,Y,Z)	JLSTL (LSTYLE)
J3MRK (X,Y,Z,MARKNO)	JLWID (LWIDTH)
J3PGN (NPOLY,NVEC,XVEC,YVEC,ZVEC)	JMCUR {no parameters}
J3PLY (NPNTS,XVEC,YVEC,ZVEC)	JNEWF {no parameters}
JALPH (ID,NCHARS,STRING)	JOESC (ID,OPCODE,ISIZE,RSIZE,ILIST,RLIST)
JASPK (XSIZE,YSIZE)	JOPEN (NAME)
JBATC {no parameters}	JPICK (ID,ECHO, <u>BUTTON</u> , <u>NAME</u> , <u>PICKID</u>)
JBEGN {no parameters}	JPICL (COLOR)
JBUTN (ID,ECHO, <u>BUTTON</u>)	JPIKP (ID,VX,VY)
JCLOS {no parameters}	JPILS (LSTYLE)
JCLPD (HITHR,YON)	JPKID (PICKID)
JCLPW (OPCODE)	JPLIM (ID,XMIN,XMAX,YMIN,YMAX)
JCLR {no parameters}	JPPMM (ID,VX,VY, <u>XMM</u> , <u>YMM</u>)
JCMOD (OPCODE)	JPROJ (OPCODE,DU,DV,DN)
JCOLM (ID, MODEL)	JPSTL (PINDEX)
JCOLR (COLOR)	JPURG (NAME)
JCORI (XBAS,YBAS,ZBAS,XPLN,YPLN,ZPLN)	JR2DR (DX,DY)
JCSIZ (WIDTH,HEIGHT,GAP)	JR2MK (DX,DY,MARKNO)
JDCOL (ID,COLOR,COLP 1,COLP 2,COLP 3)	JR2MV (DX,DY)
JDDEV (ID,CLASS)	JR2PG (NPOLY,NVEC,DXVEC,DYVEC)
JDFNT (FONT,SLANT,FNTLOC)	JR2PL (NPNTS,DXVEC,DYVEC)
JDINT (ID,WSPLEN,WSPNAM,IODLEN,IODES,CONTRL)	JR3DR (DX,DY,DZ)
JDLIM (ID,XMIN,XMAX,YMIN,YMAX)	JR3MK (DX,DY,DZ,MARKNO)
JDMOD (MATRIX)	JR3MV (DX,DY,DZ)
JDPMM (ID,VX,VY, <u>XMM</u> , <u>YMM</u>)	JR3PG (NPOLY,NVEC,DXVEC,DYVEC,DZVEC)
JDPST (ID,PINDEX,DENSTY,ORIENT,EDGE,DIDDF)	JR3PL (NPNTS,DXVEC,DYVEC,DZVEC)
JDPTH (HDIST,YDIST)	JRNAM (OLDNAM,NEWNAM)
JEDEV (ID,CLASS,IODLEN,IODES)	JRSET {no parameters}
JEND {no parameters}	JSDET (NAME,OPCODE)
JFONT (FONT)	JSERR (RLEVEL,ALEVEL)
JGDET (OPCODE)	JSHI (NAME,OPCODE)
JGHI (OPCODE)	JSLOC (ID,ECHO, <u>VX</u> , <u>VY</u>)
JGVIS (OPCODE)	JSVAL (ID,ECHO,SUBVAL, <u>VALUE</u>)
JHAND (OPCODE)	JSVIS (NAME,OPCODE)
JI1IN (OPCODE, <u>VALUE</u>)	JTEXH (NCHARS,STRING)
JI1RE (OPCODE, <u>VALUE</u>)	JTEXL (NCHARS,STRING)
JI2RE (OPCODE, <u>VAL 1</u> , <u>VAL 2</u>)	JTEXM (NCHARS,STRING)
JI3RE (OPCODE, <u>VAL 1</u> , <u>VAL 2</u> , <u>VAL 3</u>)	JUPDT {no parameters}
JI4RE (OPCODE, <u>VAL 1</u> , <u>VAL 2</u> , <u>VAL 3</u> , <u>VAL 4</u>)	JVDIS (DIST)
JICOL (ID,COLOR, <u>COLP 1</u> , <u>COLP 2</u> , <u>COLP 3</u>)	JVIEW (VPXMIN,VPXMAX,VPYMIN,VPYMAX)
JICP (<u>X</u> , <u>Y</u> , <u>Z</u>)	JVPLN (XNRM,YNRM,ZNRM,XUP,YUP,ZUP)
JIERR (<u>ERRNUM</u> , <u>LEVNUM</u> , <u>SUBNUM</u> , <u>INFO</u>)	JVREF (X,Y,Z)
JIESC (ID,OPCODE,ISIZE,RSIZE, <u>ILIST</u> , <u>RLIST</u>)	JVSAL (OPCODE)
JIMAT (OPCODE,MATRIX)	JVTOW (VX,VY, <u>X</u> , <u>Y</u> , <u>Z</u>)
JIPST (ID,INDEX,DENSTY,ORIENT,EDGE,DIDDF)	JWEND (ID)
JISGA (NAME, <u>VALUE</u>)	JWIND (DWUMIN,DWUMAX,DWVMIN,DWVMAX)
JISGW (SEGNAM,MAXNAM,COUNT, <u>IDARRY</u>)	JWLOC (ID,ECHO, <u>BUTTON</u> , <u>VX</u> , <u>VY</u>)
JITSZ (NCHARS,STRING, <u>SHIGH</u> , <u>SWIDE</u>)	JWOFF (ID)
JIVOF {no parameters}	JWON (ID)
JIVON {no parameters}	JWTOV (X,Y,Z, <u>VX</u> , <u>VY</u>)
JIWND (<u>XARRAY</u> , <u>YARRAY</u> , <u>ZARRAY</u>)	JWVAL (ID,ECHO,SUBVAL, <u>BUTTON</u> , <u>VALUE</u>)

Preface

Who needs to use this manual?

The *Advanced Graphics Package Supplement for HP-UX Systems* complements the *Advanced Graphics Package Reference Manual* and is intended for users of Hewlett-Packard's Advanced Graphics Package (AGP). The Supplement contains AGP reference information which is specific to the HP-UX operating system.

How is it organized?

The *AGP Supplement for HP-UX Systems* is divided into two parts. Part I contains HP-UX operating system-dependent topics. Part II describes those calls which are HP-UX operating system-dependent.

Calls in Part II have the same format as those in the *AGP Reference Manual*. You may insert these calls in the Reference Manual (in alphabetical order) or use the Supplement as is.

Table of Contents

Part I

System-Dependent Topics

Operating Environment.....	I-1
I/O File Descriptors.....	I-1
Programming Languages.....	I-1
FORTRAN.....	I-1
Pascal.....	I-3
C.....	I-6
Library Names.....	I-9
Linking AGP Programs.....	I-10
Linking the User Program.....	I-10
Linking the Work Station Program.....	I-11
Outspooling Graphics.....	I-12
Advanced Concepts.....	I-13
Description of AGP Programs.....	I-13
User Program Description.....	I-13
Work Station Program Description.....	I-14
Naming Conventions for AGP Routines.....	I-15
System Resources Required by AGP.....	I-15
Outside World Signals.....	I-15
AGP Program-to-Program Communications.....	I-16
Changing the Size of the Segment Display Area.....	I-16
Polygon Set Efficiency Considerations.....	I-17
Changing the Size of the Polygon Intercept Buffer.....	I-17
Changing the Size of the Polygon Vertex Buffer.....	I-17
Changing the Size of the Color Table.....	I-18
Fonts.....	I-19

Part II

System-Dependent Calls

Introduction.....	II-1
JDFNT.....	II-2
JDINT.....	II-4
JEDEV.....	II-9
JIWS.....	II-13
JSERR.....	II-23

List of Tables

AGP Parameters and Pascal Data Types.....	I-4
AGP Parameters and C Data Types.....	I-7
Device Handler File Naming Conventions.....	I-10

Part I

System-Dependent Topics

OPERATING ENVIRONMENT

AGP is supported on the HP-UX operating system on the following HP 9000 processors:

Model 20 series 500
Model 30 series 500
Model 40 series 500

I/O UNIT DESCRIPTORS

The I/O unit descriptor specified in the initialization calls is a device file. To create this file, use the `mknod` command (described in the *HP-UX Reference Manual*).

PROGRAMMING LANGUAGES

FORTRAN

Unless otherwise specified, the FORTRAN compiler defaults all integers to 32 bits. AGP requires 16 bit integers. To assign 16 bit integers, use one of the following methods:

1. Declare variables as `INTEGER*2`.
2. Use the compiler option, `$OPTION SHORT INTEGERS`.
3. Use the `-I2` option in the run string for the compiler.

Note that method 1 assigns only declared variables and methods 2 and 3 force all subsequent integers (unless declared otherwise) to 16 bits.

Characters must be passed to AGP in integer arrays or by using in-line Hollerith strings. You can do this with data statements as shown in the program example.

The file /usr/lib/graphics/fortran/fagpl.h is supplied with AGP. It contains aliases to the AGP routine names which are more English-like. To use these names, include this file in a AGP program as follows:

```
INCLUDE '/usr/lib/graphics/fortran/fagpl.h'
```

The following FORTRAN example of a program called USER is complete. You can type it into the computer and execute it.

```
$OPTION SHORT INTEGERS
PROGRAM USER
C
C..AGP user program to draw a line.
C The graphics output is hard coded to /dev/crtgraphics
C
C..Set up variables
C
      INTEGER WSLEN, WSPNAM(2), IODES(8), IODLEN, IERR, CNTRL
      DATA WSLEN/4/
      DATA WSPNAM/2HWS,2HP1/
      DATA IODES/2H/d,2Hev,2H/c,2Hrt,2Hgr,2Hap,2Hhi,2Hcs/
      DATA IODLEN/16/
C
C..Initialize the system
C
      CALL JBEGN
C
C..Initialize a work station program: work station 1, wspnam='WSP1'
C at /dev/crtgraphics with no control bits set. Then enable
C it for graphics output.
C
      CNTRL = 0
      CALL JDINT (1, WSLEN, WSPNAM, IODLEN, IODES, CNTRL)
      CALL JWON(1)
C
C..Everything between the comment lines may be changed as needed.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C..Use default attributes and viewing transformation to draw line
C diagonally across the display.
C
      CALL J2MOV (-1.0,-1.0)
      CALL J2DRW (1.0,1.0)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C..Disable the workstation, terminate it, end the AGP system, and
C terminate this program.
C
      CALL JWOFF(1)
      CALL JWEND(1)
```

```
CALL JEND
END
```

Pascal

In general, the difficulties of accessing AGP routines are the same as accessing any non-Pascal routine from a Pascal program. Therefore, a review of that topic in the *Pascal Language Manual* may be useful.

Parameters (param1, param2, ...) passed to AGP routines must be declared VAR, or call-by-reference. Note that any external routine may modify the actual parameters passed to it when declared VAR in Pascal. However, AGP does not modify input parameters. All AGP routines should be declared as external procedures as follows:

```
PROCEDURE AGP_routine
  $ALIAS 'jxxxx'$
  (VAR param1 : type1;
   VAR param2 : type2;
   :
   :
   VAR paramN : typeN);
EXTERNAL;
```

where jxxxx is the true name of the AGP routine, and AGP_routine is a convenient alias.

The following files are supplied with AGP:

```
/usr/lib/graphics/pascal/pagp1.h
/usr/lib/graphics/pascal/pagp2.h
```

These files provide standard data types and external declarations for inclusion in all application programs. The file pagp2.h contains declarations for the standard AGP routine names as well as aliases to a set of more English-like names. The contents of these files may be copied to a local directory and modified as needed.

The application program must use the data types which AGP expects. Unfortunately, Pascal's and FORTRAN's data types are not always equivalent. As a Pascal user, you should therefore declare and use appropriate data types for both the formal and actual parameters of AGP routines. One example of this incompatibility is the data type INTEGER. Pascal represents integers in 32 bits while AGP's FORTRAN code always assumes them to be 16 bits.

The following declarations cause the variable `graphics_id` to be stored in a 16 bit integer:

```
TYPE
    INT  = -32768..32767;

VAR
    graphics_id : INT;
```

Table I-1 summarizes correspondences between AGP parameters and matching Pascal data types. A file containing standard data-type declarations for Pascal is supplied with the AGP product.

Table I-1. AGP Parameters and Pascal Data Types

AGP Parameter	Corresponding Pascal Data Type
Integer	-32768..32767
Character string of n characters	PACKED ARRAY[1..n] OF CHAR or ARRAY [1..n] of CHAR
Real	REAL
Array of n integers	ARRAY [1..n] OF -32768..32767

A PASCAL version of program USER is shown below.

```

PROGRAM user; { AGP user program to draw a line }

{ The following include files are part of the AGP product }

{constant and type declarations}
$INCLUDE '/usr/lib/graphics/pascal/pagp1.h'$

VAR
  IERR : int;
  ID, CONTROL : int;
  WSPNAM, IODES : string132;
  WSPLLEN, IODLEN : int;
  XMIN, XMAX, YMIN, YMAX : real;

{external procedure declarations}
$INCLUDE '/usr/lib/graphics/pascal/pagp2.h'$

BEGIN { Main of user }

{ Initialize variables }
  ID:=1;
  WSPLLEN:=4;
  WSPNAM='WSP1';
  IODLEN := 16;
  IODES := '/dev/crtgraphics';
  XMIN := -1.0;
  XMAX := 1.0;
  YMIN := -1.0;
  YMAX := 1.0;
  CONTROL := control_spooling_off;

{ Initialize the AGP system. }
  jbegn;

{ Initialize a work station program: work station 1, }
{ wspnam='WSP1' at /dev/crtgraphics with no control bits set. }
{ Then enable it for graphics output. }
  jdint (ID WSPLLEN, WSPNAM, IODLEN, IODES, CONTROL);
  jwon (ID);
{ Everything between the dashes may be changed as needed }
{ ----- }

{ Using default attributes and viewing transformation, draw line}
{ diagonally across the display. }
  j2mov ( XMIN, YMIN );
  j2drw ( XMAX, YMAX );

{ ----- }

{ Disable the work station, terminate it, and end the AGP system}
  jwoff (ID);
  jwend (ID);

```

```
    jend;  
END.  { Main of User}
```

C

In general, the difficulties of accessing AGP routines are those which arise when accessing any FORTRAN routine from C. See the *FORTRAN Manual* for a discussion of the FORTRAN - C interface.

You must pass actual parameters (param1, param2, ...) to AGP routines by reference only. Therefore, all actual parameters passed to AGP must be pointers or variables prefixed with the address operator (&). All formal parameters returned from AGP must be pointer variables. All integer parameters should be declared as short integers and all reals as float.

AGP expects character parameters to be passed in integer arrays. As a C programmer, you can pass these in one of two ways:

1. Use an array of short integers and a character pointer to the array; assign the characters individually using the character array (see C program below).
2. Pass a double-quoted character string. For example, the jdint call in the C program could have been coded:

```
    jdint (&id,&wsplen,"WSP1",&iodlen,"/dev/crtgraphics",&control);
```

In either case, the number of characters in the string passed must be at least the length specified. For character output, AGP requires a pointer to an array of short integers.

Table I-2 summarizes correspondences between AGP parameters and matching C data types. User-supplied names are bracketed by <>.

Note: Characters are packed, two per element, in a short integer array and require $(n+1)/2$ elements to store n characters. For example, 15 characters could be stored in an array declared `short string[8]`.

Table I-2. AGP Parameters and C Data Types

AGP Parameter	Corresponding C Data Type
Integer	short
Character string of n characters (see note above)	short <arrayname> $[(n+1)/2]$
Real	float
Array of n integers	short <arrayname> $[n]$

The file `/usr/lib/graphics/c/cagp1.h` is supplied with the AGP product. It contains macro definitions of more English-like names. To use these names, include this file in a AGP program as follows:

```
$include '/usr/lib/graphics/c/cagp1.h'
```

A C version of program USER is shown below.

```
main ()
{
    /* begin main of user */

    /******
    /*      AGP user program to draw a line.
    /*      The wspnam is hard-coded to WSP1.
    /*      The graphics output is hard-coded to /dev/crtgraphics.
    /******

    short  id,wspnen,wspnam[2],iodlen,iodes[8],control,ierr;
    float  xmin,xmax,ymin,ymax;
    char    *wscnam = (char *)wspnam;
    char    *ioces = (char *)iodes;

    /******
    /*      Initialize the AGP system.
    /******

    jbegin();

    /******
    /*      Initialize a work station program;
    /*      work station 1, wspnam = 'WSP1', at /dev/crtgraphics
    /*      with no control bits set.
    /*      Then enable it for graphics output.
    /******

    id = 1;
    wspnen = 4;
    wscnam[0] = 'W';
    wscnam[1] = 'S';
    wscnam[2] = 'P';
    wscnam[3] = '1';
    iodlen = 16;
    ioces[0] = '/';
    ioces[1] = 'd';
    ioces[2] = 'e';
    ioces[3] = 'v';
    ioces[4] = '/';
    ioces[5] = 'c';
    ioces[6] = 'r';
    ioces[7] = 't';
    ioces[8] = 'g';
    ioces[9] = 'r';
    ioces[10] = 'a';
    ioces[11] = 'p';
    ioces[12] = 'h';
    ioces[13] = 'i';
    ioces[14] = 'c';
    ioces[15] = 's';
```

```

        control = 0;
        jdint(&id,&wsplen,wsplnam,&iodlen,iodes,&control);
        jwon(&id);

/*****
/*      Everything between the dashes may be changed as needed. */
/*-----*/

/*****
/*      Use default attributes and viewing transformation          */
/*      to draw a line diagonally across the display.             */
/*****

        xmin = -1.0;
        xmax = 1.0;
        ymin = -1.0;
        ymax = 1.0;
        j2mov (&xmin,&ymin);
        j2drw (&xmax,&ymax);

/*-----*/
/*      Disable the work station, terminate it, and end the AGP */
/*      system.                                                  */
/*****

        jwoff(&id);
        jwend(&id);
        jend();

}      /* end main of user */

```



LIBRARY NAMES

This supplement refers to libraries in the same way they are referenced using the -l option on the linker. By adding the appropriate prefix and suffix to these names, the files can be found in the /lib directory. For example, the file names for the UPLIB and WSPLIB libraries are /lib/libUPLIB.a and /lib/libWSPLIB.a respectively.

Files containing device handlers in AGP are found in /lib and are named so that the first character of the file name correlates with the logical function performed by the handler and the last four characters correlate with the physical device. For example, a HP model 2623A display handler would be found in /lib/libD0019.a. For a complete list of device handlers supported by AGP, refer to the *Device Handlers Manual*.

Table I-3 shows the first character used for each logical device handler.

Table I-3. Device Handler File Naming Conventions

First Character	Logical Device
A	Alphanumeric
B	Button
D	Graphics Display
K	Keyboard
L	Locator
P	Pick
V	Valuator

LINKING AGP PROGRAMS

Performing an AGP graphics task requires the following two programs:

1. User Program (UP): a device-independent application program that uses AGP subroutines to create graphical images.
2. Work Station Program (WSP): a device-dependent program that provides an interface between AGP and the various physical devices.

These programs are loaded separately and execute by using pipes to communicate. Refer to the *Description of AGP Programs* and *AGP Program-to-Program Communications* sections of this supplement for a detailed description of each of these programs.

Linking the User Program

The UP is linked by combining the application object program with the necessary routines in the AGP User Library (UPLIB). Because AGP is written in FORTRAN and C, you must search the FORTRAN, C, and math libraries for every link procedure. Some examples of typical command sequences are shown below. Note that some of the libraries are automatically searched and therefore do not have to be specified in the runstring.

1. For FORTRAN/9000:

```
fc userprogram.f -lUPLIB
```

2. For Pascal/9000:

```
pc userprogram.p -lUPLIB -lF77 -lI77
```

3. For C:

```
cc userprogram.c -lUPLIB -lF77 -lI77 -lm
```

See the appropriate compiler manual and/or ld documentation for details on additional compiler and linker options.

Linking the Work Station Program

You must make two decisions when linking a typical WSP:

1. Determine the functions and devices it is to contain.
2. Determine the WSP program name.

The process of linking a typical WSP, with an explanation of each step, is as follows:

- | | |
|---|---|
| 1. Link /usr/lib/wsp.o | This is the WSP main program. |
| 2. Search WSPLIB | This appends the Segment Display Area (SDA), the SDA management routines, and other AGP routines. |
| 3. Search one device handler file for each logical function chosen. | For example, search B0001 for the HP model 2623A button. Note that it is legal for one physical device type to support several logical functions if desired. The graphics display handler file <u>must</u> be the last device handler library searched. |
| 4. Search DIDD | This is the DGL library. The search appends all the DGL routines necessary to support the handlers. It also appends dummy handlers for any logical functions not explicitly searched in the previous step. |

5. Search the system
library

This picks up necessary system support routines (performed automatically by the compilers).

The following example shows a link command sequence:

```
ld -e _start /lib/frt0.o /usr/lib/wsp.o -lWSPLIB \  
-lK0001 -lL0004 -lD0019 -lDIDD -lF77 -lI77 -lm -lc
```

In this example, the decision was made to support only the graphics display, keyboard, and locator logical functions with the WSP. Furthermore, the HP model 2623A was chosen for the display and keyboard and the HP model 9111 data tablet for the locator.

Once loaded, the WSP is ready to be used by the UP. Its default name is a.out which is the input/output file descriptor to be used in the JDINT call to initialize this WSP.

This WSP could have been renamed by either using the -o option on the runstring or copying or moving the a.out file to another file name. In either case, the final file name is the one to be used in the JDINT call. An example of each of these is as follows:

1. `ld -e _start /lib/frt0.o /usr/lib/wsp.o -lWSPLIB \
-lK0001 -lL0004 -lD0019 -lDIDD -lF77 -lI77 -lm -lc -o wspname`

or after a.out is created:

2. `mv a.out wspname`

In either case, the WSP specified in the JDINT call is wspname. If system resources are limited, you may make the WSP not shareable.

OUTSPOOLING GRAPHICS

AGP allows you to outspool graphics commands but does not do the spooling itself.

You can enable outspooling by setting bit 0 of the control word in the JDINT call. If bit 0 is set (i.e., spooling is enabled), AGP does not check the type of I/O unit descriptor to ensure that it is correct.

If the alphanumeric device and the graphics display device are identical, you can spool alphanumeric and graphics data simultaneously. It is necessary, however, to enable the graphics display before enabling the alphanumeric display.

You can send AGP output to a disc file which can later be sent to the graphics/alphanumeric device. This is useful when identical copies of a picture are needed.

Once created, you can send the file only to a physical device which is the same type supported by the device handler linked with the WSP that was used by the applications program. For example, if a file was created using a HP model 9872S device handler, the file can only be sent to a HP model 9872S plotter.

ADVANCED CONCEPTS

Description of AGP Programs

AGP is divided into the user program (UP) and the work station program (WSP). (Refer also to the *Linking AGP Programs* section of this manual). There are several advantages to this approach.

1. The UP is device independent since it can control any supported graphics configuration by initializing the appropriate WSP.
2. The configuration of a WSP can be changed to include new or different peripherals without reloading the user program.
3. User code space is increased because some of the AGP code necessary to perform the requested graphics task is in the WSP.

USER PROGRAM DESCRIPTION

The user program (UP) contains the following components:

1. User written code: written in the host language; contains calls to AGP subroutines that specify the graphics task.
2. AGP command modules: generate virtual coordinate data from the user specified world coordinate data. Only those routines referenced by the user specified AGP calls and their support routines are loaded into the UP.
3. AGP communication module: handles communications between the AGP modules in the UP and those in the WSP.

The UP can communicate with up to four WSP's simultaneously in AGP. Each WSP drives its own set of devices.

WORK STATION PROGRAM DESCRIPTION

The WSP is a program that receives commands and virtual coordinate data from the AGP command modules and performs the graphics tasks specified in the commands. You can think of it as a logical graphics device that performs the following graphics tasks as requested by the UP:

1. Maintains the necessary data structures to perform all actions that may be requested of it by the UP.
2. Contains a set of device handlers that perform all I/O to a specified set of physical graphics devices.

The devices supported by a fully equipped WSP are:

Alphanumeric

Button

Graphics display

Keyboard

Locator

Pick

Valuator

The WSP contains the following components:

1. Communication and control module: communicates with the AGP modules in the UP and calls the appropriate AGP modules in the WSP to process all requests from the UP.
2. Segment display area: a data structure used to store information about graphics segments (e.g., the primitives comprising a segment).
3. Segment display area manager: consists of all AGP code used to process and inquire about the segment display area (e.g., the code used to perform software picking and to redraw the picture).
4. Device handler code: consists of one device handler for each logical AGP device.

Naming Conventions for AGP Routines

AGP internal routines and common blocks start with the characters j, i, q, m, c, and f followed by a maximum of four other characters or numbers. To avoid naming conflicts, you should not name any block datas, common blocks, subroutines, or functions with these beginning letters.

System Resources Required by AGP

On all supported systems, each UP requires two file descriptors for the interprocess communication channels for each WSP. During interprocess channel setup (JDINT), four file descriptors are needed for the two pipe calls that must be made to establish communications between the UP and the WSP. The WSP uses the standard in and standard out file descriptors for its communications channel; it also uses the standard error for reporting its error conditions. These file descriptors (0, 1, and 2 respectively) must therefore not be closed before the JDINT call is completed. If system resources are limited, you may make the WSP and the UP not shareable.

AGP ensures that all system resources it uses are properly returned to the system upon program termination. The UP and the WSP return all the resources they are not currently using to the system. Upon process termination, all remaining system resources are returned to the HP-UX operating system.

Outside World Signals

A graphics application can be terminated with any of the following signals sent to the UP:

- Hang up - signal #1
- Interrupt - signal #2
- Quit - signal #3
- Software Termination - signal #15

AGP can only use these signals if they are set to perform system (HP-UX) default action. Since programming languages often catch signals and change them, the application program may have to specifically set signals to system default action before the JBEGN call.

When the UP receives one of the signals, it reports AGP error 118 - level 4, and starts each of the WSPs into an orderly termination before terminating itself and graphics. A WSP will attempt to restore its devices to a "ready" state. If a WSP is hung on a device (e.g., has a pending read on a powered down device), it will still terminate.

Graphics relies on the UP for controlling the WSP's reaction to signals. Therefore, all signals should be sent to the UP. In the event that a WSP is sent a signal directly and terminates, the UP reports a communication error (AGP error 106) and continues.

Note: For WSPs with devices that store commands in internal buffers (such as the HP 7580B Drafting Plotter), graphics output may continue for a while after the Alarm Clock signal has been sent. This delay is because the WSP is waiting for its devices to clear their buffers so that it can leave them in a normal state.

If the UP intercepts the signals, instead of AGP, the program can terminate AGP and the WSPs with a JEND call. If a WSP is hung on a device, the UP may have to circumvent the normal termination route and send an Alarm Clock signal to the WSP's process id to tell the WSP to terminate itself and leave the device in a "ready" state. It is possible for the UP to report an AGP error 0 when it first tries to communicate with a signal-terminated WSP.

AGP Program-to-Program Communications

Program-to-program communication through interprocess channels, called pipes, allows the UP and WSP to communicate with each other concurrently. AGP's use of pipes is transparent to you because subroutines handle the allocation and de-allocation of the file descriptors used by the pipes.

Two pipes with two file descriptors each are established for interprocess communication when a WSP is initialized. The UP's input file descriptor is mapped to the WSP's standard output file descriptor, and vice-versa, leaving two file descriptors free to be returned to the HP-UX system. The UP uses its output file descriptor to transmit messages to the WSP. The WSP uses its input file descriptor to receive the message (two ends of the same pipe). In a similar manner, the WSP uses its other file descriptor to transmit messages to the UP. The HP-UX system cleans up these file descriptors upon program termination.

Changing the Size of the Segment Display Area

You can increase the size of the Segment Display Area (SDA) to the maximum allowed by AGP at the link time of a WSP. The module `/usr/lib/k2sdf.o` contains the enlarged data area for the SDA buffer. The default value is 4,000 words. When `/usr/lib/k2sdf.o` is linked into the WSP, this value becomes 524,272 words. The `chatr +z` command option is recommended after linking so that the WSP is demand loadable.

To increase the size of the SDA in the WSP, link the module `/usr/lib/k2sdf.o` with the WSP. Two examples are:

1. For the link editor, ld:

```
ld -e _start /lib/frt0.o /usr/lib/wsp.o /usr/lib/k2sdf.o \  
-lWSPLIB -lK0001 -lD0019 -lDIDD -lF77 -lI77 -lm -lc  
chatr +z a.out
```

2. For FORTRAN/9000:

```
fc /usr/lib/wsp.o /usr/lib/k2sdf.o -lWSPLIB -lK0001 \  
-lL0004 -lD0019 -lDIDD  
chatr +z a.out
```

Polygon Set Efficiency Considerations

CHANGING THE SIZE OF THE POLYGON INTERCEPT BUFFER

As each interior fill line is computed, the intersections of the fill line with the polygon set boundary are stored in the DGL part of the WSP, i.e., intersection coordinates are stored in an intercept buffer. If a polygon set edge has *n* sides, then up to *n* intercepts could be generated. The worst case would be a concave or self-intersecting polygon set. If the number of intercepts overflows the buffer, the fill line is not drawn. The intercept buffer has a 100 point default capacity which can be increased.

You can change the size of the polygon intercept buffer as follows:

1. Make a copy of the source file /usr/lib/graphics/fortran/tlint.f.
2. Modify the copy using the instructions documented in the source code.
3. Compile the modified copy of the source module, e.g., `fc -c tlint.f`.
4. Link the new polygon intercept buffer with the WSP. The following example is shown for the FORTRAN compiler:

```
fc /usr/lib/wsp.o tlint.o -lWSPLIB -lB0004 -lL0006 -lD0019 -lDIDD
```

CHANGING THE SIZE OF THE POLYGON VERTEX BUFFER

The vertices of a polygon set are stored in the vertex buffer of the WSP. The default buffer holds up to 100 vertices. If, at anytime after clipping, more vertices are generated than a work station's vertex buffer can hold, Error 57 is generated and the polygon set is not filled on that work station.

You can change the size of the polygon vertex buffer as follows:

1. Make a copy of the source file `/usr/lib/graphics/fortran/klfil.f` .
2. Modify a copy using the instructions documented in the source code.
3. Compile the modified copy of the source module, e.g., `fc -c klfil.f` .
4. Link the new polygon vertex buffer in with the WSP. The following example is shown for the FORTRAN compiler:

```
fc /usr/lib/wsp.o klfil.o -lWSPLIB -lB0004 -lL0006 -lD0019 -lDIDD
```

Changing the Size of the Color Table

Every graphics display device has a default color table which is loaded whenever the corresponding device handler is loaded. The default tables of specific devices are given in the *Device Handlers Manual*.

For some devices, the number of colors in the default table can be increased at link time. Consult the *Device Handlers Manual* for a specific device to determine if its color table can be changed. This number must not be made greater than 32767 (the graphics system limit) or less than the default size for the device.

If the number of entries in the color table is increased, each new entry will have the same default parameters as the first entry. For example, if the default table has 10 entries and its size is increased by five, then entries 11-15 will have the same color definition as the first entry. If nothing more is done, the effect of referencing entry 12 will be the same with the modified table as with the default table. If the new entries permit the display of new colors, their contents must be redefined within the program using JDCOL.

You can change the size of the color table as follows:

1. Make a copy of the source file `/usr/lib/graphics/fortran/zlctb.f` .
2. Modify the copy using the instructions documented in the source code.
3. Compile the modified copy of the source module, e.g., `fc -c zlctb.f` .
4. Link the new color table in with the WSP. The following example is shown using the FORTRAN compiler:

```
fc /usr/lib/wsp.o zlctb.o -lWSPLIB -lB0004 -lL0006 -lD0019 -lDIDD
```

Fonts

The following is a listing of the integer (FNTLOC) representing the font style for each of the six high quality text fonts (stored in memory) which are supplied with AGP:

<u>FNTLOC</u>	<u>Font style</u>
1	Eurostyle
2	Simplex Roman
3	Triplex Roman
4	Script
5	Mathematical
6	Gothic

Refer to the *AGP Reference Manual* for a detailed description of each font style.



Part II

System-Dependent Calls

INTRODUCTION

The calls described in this supplement contain system-dependent information. These descriptions, together with the (system-independent) call descriptions in the *AGP Reference Manual*, provide complete documentation for AGP subroutines.

Since the format of these calls is the same as those in the *AGP Reference Manual*, you can insert them (in alphabetical order) in that manual or use this manual as it is.

JDFNT

JDFNT

PURPOSE: To associate a font number with a font definition for use with high quality text.

CALLING SEQUENCE: CALL JDFNT (FONT,SLANT,FNTLOC)

FONT	[INTEGER; Input] FONT number to be associated with the FONT definition specified. Used in the JFONT call to select a desired font. (1 <= FONT <= 16)
SLANT	[REAL; Input] Angle in radians specifying the character slant. Range is from $-\pi/2.0$ to $+\pi/2.0$ with a value of 0.0 specifying no slant. ($-\pi/2.0 \leq \text{SLANT} \leq \pi/2.0$).
FNTLOC	[INTEGER; Input] Location of the desired font. JDFNT associates FONT with the specified slant and font definition. See the <i>Font</i> section in this manual for the integer identifying each font style in memory. (1 <= FNTLOC <= number of fonts in memory)

Whenever the font number is referenced for subsequent high quality text output, the character data in the specified font definition will be used. Note that a font must be defined (JDFNT) and then set to the current font (JFONT) before it can be used for high quality text.

SLANT is used to generate slanted text. The major use of this parameter is to generate italicized text. The angle specified must be between $-\pi/2.0$ and $+\pi/2.0$. A value of 0.0 for SLANT will generate characters that are not slanted.

ERROR CONDITIONS:

064 Level 0: AGP was not initialized by JBEGN

INFO:	Undefined.
ACTION:	An implicit call to JBEGN is made.

090 Level 3: The font number specified is out of range. Possibly a non-positive number was passed.

INFO: The font requested.
ACTION: Call ignored.

107 Level 3: The FNTLOC specified is less than 1 or greater than the number of fonts in memory. Possibly a negative or real number was passed.

INFO: FNTLOC specified.
ACTION: Call ignored.

089 Level 2: The slant is out of range.

INFO: Undefined.
ACTION: A value of 0.0 is used.



JDINT

JDINT

PURPOSE: To initialize a work station.

CALLING SEQUENCE: CALL JDINT(ID,WSPLEN,WSPNAM,IODLEN,IODES,CONTRL)

ID	[INTEGER; Input] The ID used to refer to this work station in subsequent AGP calls. (1 <= ID <= 4)
WSPLEN	[INTEGER; Input] The number of characters from WSPNAM to be used as the work station program name.
WSPNAM	[INTEGER; Array; Input] The character string (in packed ASCII format) containing the name of the work station program.
IODLEN	[INTEGER; Input] The number of characters from IODES to be used as the I/O unit descriptor associated with the graphics display of the work station.
IODES	[INTEGER; Array; Input] The character string (in packed ASCII format) containing the I/O unit descriptor.
CONTRL	[INTEGER; Input] The number used to control the characteristics of the work station. The following bit map shows how bits should be set (set all unused bits to zero).

```
-----  
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | X | 0 | 0 | X |  
-----  
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

BIT 0 = 0 - No system spooling will be possible to the I/O file unit.

= 1 - I/O unit descriptor may be outspooled using system spooling.

As part of the work station initialization process, AGP verifies (if possible) that the device at the I/O unit descriptor is of the proper type for the work station program named in WSPNAM. To do this, it sends an inquiry to which the device responds with its particular identification sequence. Setting bit 0 inhibits the inquiry process. AGP does not perform spooling. Input will not be possible when outspooling is in use, so setting this bit causes all subsequent input requests from the same device to be flagged with errors and ignored.

BIT 1-2 - Currently unused. Set to 0.

BIT 3 = 0 - Simulated raster erase will not be enabled.

= 1 - Simulated raster erase will be enabled. Some graphics output devices have the ability to "undraw" images. AGP will use this feature when the simulated raster erase bit is set in the JDINT call. This feature allows AGP to do such things as removing purged segments without clearing the graphics display device and redrawing all visible segments. Note: The "undrawing" is done by redrawing the image in the background color. Therefore, lines crossing the image to be purged may also have parts of them undrawn thus leaving "holes". See the *Device Handlers Manual* for the effect of bit 3 on a particular device.

BITS 4-6 - Currently unused. Set to 0.

BIT 7 = 0 - Immediate erase. AGP will clear the graphics display during initialization.

= 1 - No erase. Whatever was on the display stays there. This action is device dependent. See the *Device Handlers Manual* for the effect of bit 7 on a particular device.

BITS 8-15 - Currently unused. Set to 0.

JDINT initializes a work station. A work station must be initialized before the application program can reference it for any purpose. During this call, an attempt will be made to assure that the device at the I/O unit descriptor specified is really the type of display device that is loaded into the specified WSP. In some cases it is impossible to determine whether or not the device is the correct type. Therefore, the errors that refer to whether the device is correct may not be reported. Furthermore, when the spooling bit is set, no checks are made as to the validity of the I/O unit descriptor.

JDINT

Unpredictable results may occur if there is no device at the I/O unit descriptor, if the device is not the intended graphics device, or if the device is turned off or in a local mode.

If the graphics device is a dummy device handler, the control word (CONTRL) will be ignored.

If the graphics device is not outspooled, the device name is set to the name of the device as specified in the *Device Handlers Manual*. If the graphics output device has the outspooling bit set, the name is set to the last four characters of the display library followed by two blanks. For example, the name for the HP 2623 graphics display device handler would be "0019 ".

AGP cannot verify that the program specified is a work station program. If it is not, AGP errors will occur when the user program attempts to communicate with it.

No other work station or program should send any I/O to the device specified by the I/O unit descriptor while it is initialized by this work station. There is no AGP error check for the above condition and unpredictable results may occur if it is violated.

Note that AGP makes an implicit call to JMCUR as part of this call.

Before an application program can use one of AGP's logical devices, it must first be enabled. Each logical device can be enabled and disabled independently of the others. A call to JWON enables graphics output; a call to JWOFF disables graphics output. Similarly, a call to JEDEV selectively enables one of AGP's other logical devices; a call to JDDEV disables it. A user program may initialize up to four work stations at any one time.

The following example demonstrates initialization and termination of a work station. It also shows how one enables an input device, in this case the locator.

CALL JDINT(1,...,0)	*Initialize the work station.
CALL JWON(1)	*Enable graphical output.
:	
CALL JEDEV(1,4,...)	*Initialize the locator.
:	
{do locator input}	
:	
CALL JDDEV(1,4)	*Terminate the locator.
CALL JWOFF(1)	*Disable graphical output.
CALL JWEND(1)	*Terminate the work station.

ERROR CONDITIONS:

064 Level 0: AGP was not initialized by JBEGN.

INFO: Undefined.
ACTION: An implicit call to JBEGN is made.

074 Level 3: Work station ID out of range.

INFO: ID of the work station.
ACTION: Call ignored.

098 Level 3: WSLEN specified was less than one or greater than 132.
Possibly a negative or real number was passed.

INFO: WSLEN given.
ACTION: Call ignored.

103 Level 3: IODLEN specified was less than one or greater than 132.
Possibly a negative or real number was passed.

INFO: IODLEN given.
ACTION: Call ignored

096 Level 3: The ID number specified has been used previously for
another work station.

INFO: ID number specified.
ACTION: Call ignored.

009 Level 2: This routine should not be called while a segment is
open. Possibly a call to JCLOS was omitted.

INFO: Name of the open segment.
ACTION: An implicit call is made to JCLOS.

003 Level 3: This routine should not be called during a batch of up-
dates. Possibly missing a JUPDT.

INFO: Undefined.
ACTION: An implicit call is made to JUPDT.

115 Level 3: Specified WSP program not found. Possibly access is not
permitted or file does not exist.

INFO: ID of the work station.
ACTION: Call ignored.

JDINT

092 Level 3: The specified work station program cannot be scheduled.

INFO: ID of the work station.

ACTION: Call ignored.

149 Level 3: Rev code mismatch between the UP and the WSP (e.g., the UP was loaded with new software and the WSP was loaded with old software).

INFO: ID of the work station.

ACTION: Call ignored.

105 Level 3: The IODES specified does not match the device driver in the WSP or the device at the IODES is down. Possibly the wrong handler was loaded when the WSP was created or the wrong IODES was specified.

INFO: ID of the work station.

ACTION: Call ignored.

148 Level 3: The requested work station program is not an AGP work station program.

INFO: ID of the work station.

ACTION: Call ignored.

JEDEV

PURPOSE: To enable a logical device (other than the graphics display).

CALLING SEQUENCE: CALL JEDEV (ID,CLASS,IODLEN,IODES)

ID	[INTEGER; Input] The ID of the work station whose logical device is to be enabled. (1 <= ID <= 4)
CLASS	[INTEGER; Input] The class of the logical device which is being enabled: CLASS = 1 - Alphanumeric device. = 2 - Button device. = 3 - Keyboard device. = 4 - Locator device. = 5 - Valuator device. = 6 - Pick device.
IODLEN	[INTEGER; Input] The number of characters from IODES to be used as the I/O unit descriptor associated with the physical device.
IODES	[INTEGER; Array; Input] The character string containing the I/O unit descriptor associated with the physical device.

JEDEV enables a logical device on a particular work station. The logical device should be enabled before it is used (e.g., the pick device must be enabled (JEDEV) before it can be used for input (JPICK)). A logical device is terminated through a call to JDDEV. The work station on which the logical device is being enabled should be initialized (see JDINT). A logical device is enabled and disabled independently of the graphics output being enabled and disabled (see JWON, JWOFF).

When enabling the locator or pick device, the locator or pick echo reference position is set to its default value (see JLOCP or JPIKP).

The logical input device will not be enabled if the I/O unit descriptor is the same physical device as a graphics display device which is being outspooled (see JDINT).

JEDEV

If the specified locator/pick is not the same physical device as that of an initialized graphics display device, the logical locator/pick limits will be set to the default values for the particular locator used. If the specified locator/pick is the same physical device as the graphics display device, then the logical locator/pick limits are set to the current view surface limits.

Not all logical devices may be available on all work stations. The actual logical devices that are available on a particular work station are determined at the time the Work Station Program (WSP) is linked. JIWS can be used to inquire if a logical device is supported by a particular work station. The I/O unit descriptor and the device will be verified by AGP and the name of the device will be set accordingly. This information can be inquired via JIWS.

No other work station or program should send any I/O to the device specified by the descriptor while it is initialized by this work station. There is no AGP error check for the above condition and unpredictable results may occur if it is violated.

The following example demonstrates enabling, using, and terminating the locator.

CALL JDINT(3,...)	*Initialize work station #3.
CALL JWON(3)	*Enable graphical output.
:	
CALL JWOFF(3)	*Disable graphical output.
:	
CALL JEDEV(3,4,...)	*Enable the locator.
CALL JWLOC(3,1,BUTTN,VX,VY)	*Do input.
:	
AALL JWON(3)	*Enable graphical output.
:	
CALL JWLOC(3,1,BUTTN,VX,VY)	*Do input.
CALL JDDEV(3,4)	*Terminate the locator.
:	
CALL JWOFF(3)	*Disable graphical output.
CALL JWEND(3)	*Terminate work station.

ERROR CONDITIONS:

064 Level 0: AGP was not initialized by JBEGN.

INFO:	Undefined.
ACTION:	An implicit call to JBEGN is made.

074 Level 3: Work station ID out of range.

INFO: ID of the work station.
ACTION: Call ignored.

012 Level 3: The logical class is out of range.

INFO: Logical class requested.
ACTION: Call ignored.

103 Level 3: IODLEN specified was less than one or greater than 132.
Possibly a negative or real number was passed.

INFO: IODLEN given.
ACTION: Call ignored.

071 Level 3: The work station specified has not been initialized by JDINT. Possibly there is a call to JDINT missing or the ID is not identical to that used in the JDINT call.

INFO: ID of the work station.
ACTION: Call ignored.

003 Level 3: This routine should not be called during a batch of updates. Possibly missing a JUPDT.

INFO: Undefined.
ACTION: An implicit call is made to JUPDT.

121 Level 2: Attempt to enable a logical device that is already enabled at ID.

INFO: ID of the work station.
ACTION: Previously enabled logical device is disabled with an implicit call to JDDEV.

005 Level 3: The work station specified has a dummy device handler loaded for that logical device. An inquiry (see JIWS) of the work station may be made to determine its capabilities.

INFO: ID of the work station.
ACTION: Call ignored for that work station.

080 Level 3: Attempt to enable an input function on an outspooled device. Input can only be performed from a physical device other than that of an outspooled graphical display.

INFO: ID of the work station.
ACTION: Call ignored.

JEDEV

105 Level 3: Either the IODES specified does not match the device handler in the WSP or the device at the IODES is not working. Possibly the wrong handler was loaded when the WSP was created or the wrong IODES was specified.

INFO: ID of the work station.

ACTION: Call ignored.

JIWS

PURPOSE: To inquire about some characteristic of a specific work station.

CALLINGEG: CALL JIWS(ID,OPCODE,SSIZE,ISIZE,RSIZE,SLIST,ILIST,RLIST)

ID	[INTEGER; Input] The ID of the work station to which the inquiry is to be directed. (1 <= ID <= 4)
OPCODE	[INTEGER; Input] The code specifying which characteristic of the work station is to be queried. The supported values of OPCODE are listed below.
SSIZE	[INTEGER; Input] Maximum number of characters that are to be returned in the array SLIST by the function specified by OPCODE.
ISIZE	[INTEGER; Input] The number of integer parameters to be returned in ILIST by the function specified by OPCODE.
RSIZE	[INTEGER; Input] The number of real parameters that are returned in the array RLIST by the function specified by OPCODE.
SLIST	[INTEGER; Array; Output] An array of characters that represents characteristics of the work station specified by the value of OPCODE. Must be dimensioned large enough to hold SSIZE characters.
ILIST	[INTEGER; Array; Output] An array of integer values that represents characteristics of the work station specified by the value of OPCODE. Must be dimensioned large enough to hold ISIZE integers.
RLIST	[REAL; Array; Output] An array of real values that represents characteristics of the work station specified by the value of OPCODE. Must be dimensioned large enough to hold RSIZE reals.

JIWS allows the application programmer to inquire several characteristics about the specified work station. All characters returned are in Packed ASCII format.

JIWS

The OPCODE parameter is an integer value containing three to five digits. The most significant digits have the following meanings:

Ten thousands digit: 1 - if a string is returned in SLIST.
 0 - no string returned.

Thousands digit: The number of integer values returned in ILIST.

Hundreds digit: The number of real values returned in RLIST.

JIWS may only be called if the work station is initialized.

JIWS implicitly makes the picture current before making the inquiry.

The following codes are available:

OPCODE MEANING

201	Virtual coordinate of last PICK point. (Returns (0.,0.) if no PICK has been performed).
	RLIST(1) = X virtual coordinate. (2) = Y virtual coordinate.
247	Resolution of pick device.
	RLIST(1) = Resolution in X direction (points/mm). (2) = Resolution in Y direction (points/mm).
248	Maximum dimensions of pick device.
	RLIST(1) = Maximum size in X direction (mm). (2) = Maximum size in Y direction (mm).
249	Current pick echo position.
	RLIST(1) = X virtual coordinate. (2) = Y virtual coordinate.
250	For internal use only.
251	Marker size in virtual coordinates.
	RLIST(1) = Marker width in virtual coordinates. (2) = Marker height in virtual coordinates.
252	Resolution of graphics display.
	RLIST(1) = Resolution X direction (points/mm). (2) = Resolution Y direction (points/mm).

- 253 Maximum dimensions of graphics display.
- RLIST(1) = Maximum size in X direction (mm).
 (2) = Maximum size in Y direction (mm).
- 254 Aspect ratios.
- RLIST(1) = Aspect ratio of the virtual coordinate system.
 (2) = Aspect ratio of the logical display limits.
- 255 Resolution of locator device.
- RLIST(1) = Resolution in X direction (points/mm).
 (2) = Resolution in Y direction (points/mm).
- 256 Maximum dimensions of locator device.
- RLIST(1) = Maximum size in X direction (mm).
 (2) = Maximum size in Y direction (mm).
- 257 Current locator echo position.
- RLIST(1) = X virtual coordinate.
 (2) = Y virtual coordinate.
- 258 For internal use only.
- 259 For internal use only.
- 450 For internal use only.
- 451 For internal use only.
- 1001 For internal use only.
- 1002 Has the open segment overflowed?
- ILIST(1) = -1 - No open segment.
 (1) = 0 - SDA has not overflowed.
 (1) = 1 - The open segment has overflowed.
- 1003 Can the graphics device simulate raster erase?
- ILIST(1) = -1 - No.
 (1) = 0 - Yes, but it is not enabled.
 (1) = 1 - Yes, and it is enabled.

- 1050 Does the graphics display device support hardware clipping at physical limits?
- ILIST(1) = 0 - No.
 (1) = 1 - Yes, at the view surface boundaries.
 2 - Yes, but only to the physical limits of the device.
- 1051 Justification of view surface within the logical display limits.
- ILIST(1) = 0 - View surface is centered within logical display limits.
 (1) = 1 - View surface is positioned in the lower-left corner of the logical display limits.
- 1052 For internal use only.
- 1053 Total number of non-dithered colors supported on the graphics display. The number returned does not include the background color. (Compare opcodes 1053, 1054, and 1074).
- ILIST(1) = Number of distinct colors supported (color palette).
- 1054 Number of non-dithered distinct colors that can appear on graphics display at one time. The number returned does not include the background color. (Compare opcodes 1053, 1054, and 1075).
- ILIST(1) = Number of distinct colors that can appear on display device at one time (color gamut).
- 1055 For internal use only.
- 1056 Linestyles supported by the graphics display.
- ILIST(1) = Number of linestyles supported.
- 1057 Linewidths supported by the graphics display.
- ILIST(1) = Number of linewidths supported.
- 1058 Hardware character sizes supported by the graphics display.
- ILIST(1) = Number of character sizes supported.
 (1) = -1 Continuous character sizes.
- 1059 Number of markers supported by graphics display.
- ILIST(1) = Number of distinct markers supported.

- 1060 For internal use only.
- 1061 For internal use only.
- 1062 For internal use only.
- 1063 For internal use only.
- 1064 For internal use only.
- 1065 Number of polygon styles supported on the graphics display.
 ILIST(1) = Number of polygon styles.
- 1066 For internal use only.
- 1067 For internal use only.
- 1068 Number of polygon vertices supported by the display device.
 ILIST(1) = Number of vertices.
 If ILIST(1) = 0 - no hardware support.
 If ILIST(1) = MAXINT - no hardware limit on number of vertices.
- NOTE: MAXINT is the maximum integer size
 (CPU dependent).
- 1069 Indicates whether or not the display device supports immediate
 retroactive change of polygon style.
 ILIST(1) = 0 - No
 = 1 - Yes
- 1070 Indicates whether or not the display device supports hardware
 generation of polygons.
 ILIST(1) = 0 - No hardware support.
 = 1 - Inexact hardware support.
 = 2 - Exact hardware support.
- 1071 Indicates whether or not device supports immediate retroactive
 color change.
 ILIST(1) = 0 - No
 1 - Yes
- 1072 Indicates whether or not background color can be redefined.
 ILIST(1) = 0 - No
 = 1 - Yes

- 1073 Indicates whether or not device supports modification to color table.
- ILIST(1) = 0 - No
= 1 - Yes
- 1074 Indicate which color model is in use.
- ILIST(1) = 1 - RGB
= 2 - HSL
- 1075 Number of entries in the color table. The number returned does not include the background color. (Compare opcodes 1053, 1054, and 1075).
- ILIST(1) = 0 - The color table cannot be changed.
 > 0 - Number of entries in the table.
- 1076 Current polygon interior linestyle.
- ILIST(1) = Current polygon interior linestyle index.
- 2001 WSP Process ID.
- ILIST(1) = Upper 16 bits of the process ID.
 (2) = Lower 16 bits of the process ID.
- 2050 For internal use only.
- 3002 Segment display area.
- ILIST(1) = Usable memory resident SDA size in nodes.
 (2) = Number of unused nodes.
 (3) = Node size (number of 16 bit integers it contains).
- 11049 Pick device association.
- SLIST(1) = Character string containing device I/O unit descriptor.
ILIST(1) = Number of characters in the device I/O unit descriptor (132 characters maximum).
- 11050 Graphics display device association.
- SLIST(1) = Character string containing device I/O unit descriptor.
ILIST(1) = Number of characters in the device I/O unit descriptor (132 characters maximum).

11051 Keyboard device association.

SLIST(1) = Character string containing device I/O unit descriptor.

ILIST(1) = Number of characters in the device I/O unit descriptor (132 characters maximum).

11052 Locator device association.

SLIST(1) = Character string containing device I/O unit descriptor.

ILIST(1) = Number of characters in the device I/O unit descriptor (132 characters maximum).

11053 Button device association.

SLIST(1) = Character string containing device I/O unit descriptor.

ILIST(1) = Number of characters in the device I/O unit descriptor (132 characters maximum).

11054 Alphanumeric device association.

SLIST(1) = Character string containing device I/O unit descriptor.

ILIST(1) = Number of characters in the device I/O unit descriptor (132 characters maximum).

11055 Valuator device association.

SLIST(1) = Character string containing device I/O unit descriptor.

ILIST(1) = Number of characters in the device I/O unit descriptor (132 characters maximum).

12050 Graphics display device information.

SLIST(1) = Character string containing device name, blank-filled.

ILIST(1) = Number of characters in the device name (always 6).

(2) = Status.

= -1 There is no graphics display device loaded.

= 0 Graphics display device is not enabled.

= 1 Graphics display device is enabled.

12051 Keyboard device information.

SLIST(1) = Character string containing device name,
blank-filled.

ILIST(1) = Number of characters in the device name (always 6).

(2) = Status.

= -1 No keyboard device loaded.

= 0 Keyboard device is not enabled.

= 1 Keyboard device is enabled.

13049 Pick device information.

SLIST(1) = Character string containing device name,
blank-filled.

ILIST(1) = Number of characters in the device name (always 6).

(2) = Status.

= -1 No pick loaded.

= 0 Pick device is not enabled.

= 1 Pick device is enabled.

(3) = Number of buttons on the pick device (0 if there is
no pick device loaded).

13052 Locator device information.

SLIST(1) = Character string containing device name,
blank-filled.

ILIST(1) = Number of characters in the device name (always 6).

(2) = Status

= -1 No locator device loaded.

= 0 Locator device is not enabled.

= 1 Locator device is enabled.

13053 (3) = Number of buttons on the locator device (0 if there
is no locator device loaded). Button device
information.

SLIST(1) = Character string containing device name,
blank-filled.

ILIST(1) = Number of characters in the device name (always 6).

(2) = Status.

= -1 No button device loaded.

= 0 Button device is not enabled.

= 1 Button device is enabled.

(3) = Number of buttons on the button device (0 if there
is no button device loaded).

14054 Alphanumeric device information.

SLIST(1) = Character string containing device name,
blank-filled.
ILIST(1) = Number of characters in the device name (always 6).
(2) = Status.
= -1 No alphanumeric device loaded.
= 0 Alphanumeric device is not enabled.
= 1 Alphanumeric device is enabled.
(3) = Maximum number of lines displayed at once.
(4) = Maximum number of characters per line; -1 if un-
determinable (e.g., lineprinter).

14055 Valuator device information.

SLIST(1) = Character string containing device name,
blank-filled.
ILIST(1) = Number of characters in the device name (always 6).
(2) = Status.
= -1 No valuator loaded.
= 0 Valuator device is not enabled.
= 1 Valuator device is enabled.
(3) = Number of buttons on the valuator device (0 if
there is no valuator device loaded).
(4) = Number of sub-valuators supported.

ERROR CONDITIONS:

064 Level 0: AGP was not initialized by JBEGN.

INFO: Undefined.
ACTION: An implicit call to JBEGN is made.

023 Level 1: ID out of range.

INFO: ID of the work station.
ACTION: Call ignored.

070 Level 1: The work station specified has not been initialized.
Possibly a call to JDINT is missing or the ID is not
identical to that used in the JDINT call.

INFO: ID of the work station.
ACTION: Call ignored.

055 Level 1: The requested opcode is not available. Possibly a negative or real number was passed.

INFO: Opcode requested.
ACTION: Call ignored.

104 Level 1: SSIZE is not large enough to receive the character string requested or should be zero.

INFO: SSIZE given.
ACTION: Call ignored.

054 Level 1: ISIZE is not equal to the number of integer parameters to be returned. Possibly a negative or real number was passed.

INFO: ISIZE given.
ACTION: Call ignored.

059 Level 1: RSIZE is not equal to the number of real parameters to be returned. Possibly a negative or real number was passed.

INFO: RSIZE given.
ACTION: Call ignored.

JSERR

PURPOSE: To set the error-handling conditions.

CALLING SEQUENCE: CALL JSERR (RLEVEL,ALEVEL)

RLEVEL	[INTEGER; Input] The minimum severity error level that will be reported. (0 <= RLEVEL <= 4)
ALEVEL	[INTEGER; Input] The minimum severity error level that results in AGP termination. (0 <= ALEVEL <= 4).

JSERR sets the error handling conditions of reporting and AGP termination and implicitly makes the picture current. On the HP-UX Operating System, errors are reported to *Standard Error*. JSERR is useful when the application program is using programmatic error recovery.

JSERR allows the application program to determine which errors will be reported. All errors of severity level greater than or equal to the error report level (RLEVEL) will be reported. Initially the error report level equals 0, i.e., all errors will be reported.

If the error report level is set to 4, only level 4 errors will be logged. If this feature is used before a program is completely debugged, important information may be missed because the error messages will not be printed.

Setting ALEVEL allows the application program to recover from certain AGP detected errors that do not exceed the specified severity level. Errors greater than or equal to ALEVEL will terminate graphics. ALEVEL = 0 implies all errors and warnings are fatal; ALEVEL = 4 implies no recoverable errors are fatal. At AGP initialization, ALEVEL is set to 4.

After AGP has aborted graphics, due to an error equal to or exceeding ALEVEL, all AGP calls will be ignored except for JBEGN, JIERR, and JEND.

The last AGP error can be inquired at any time by the JIERR call.

The severity level and recovery action (if any) for each error is discussed under the "Error Condition" subheading of each subroutine description in the *AGP Reference Manual* and in this supplement. Appendix A in the *AGP Reference Manual* lists all errors, their severity levels, the subroutines where they are reported, and the recovery performed when that error occurs.

JSERR

ERROR CONDITIONS:

064 Level 0: AGP was not initialized by JBEGN.

INFO: Undefined.

ACTION: An implicit call to JBEGN is made.

043 Level 1: The report level is out of range. Possibly a real or negative number was passed.

INFO: Level requested.

ACTION: Call ignored.

044 Level 1: The abort level is out of range. Possibly a real or negative number was passed.

INFO: Level requested.

ACTION: Call ignored.

READER COMMENT SHEET

Advanced Graphics Package

Supplement for HP-UX Systems

97085-90001 June 1983

We welcome your evaluation of this manual. Your comments and suggestions help us to improve our publications. Please use additional pages if necessary.

Is this manual technically accurate? Yes [] No [] (If no, explain under Comments, below.)

Are the concepts and wording easy to understand? Yes [] No [] (If no, explain under Comments, below.)

Is the format of this manual convenient in size, arrangement and readability? Yes [] No [] (If no, explain or suggest improvements under Comments, below.)

Comments:

Date: _____

FROM:

Name _____

Company _____

Address _____

FOLD

FOLD



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070 CUPERTINO, CALIFORNIA

POSTAGE WILL BE PAID BY ADDRESSEE

Publications Manager
Hewlett-Packard Company
Engineering Productivity Division
11000 Wolfe Road
Cupertino, California 95014

FOLD

FOLD





MANUAL PART NO. 97085-90001
Printed in U.S.A.
E0683

HEWLETT-PACKARD COMPANY
Engineering Productivity Division
11000 Wolfe Road
Cupertino, California 95014